ECE482 — Introduction to Operating Systems

Homework 7

Manuel — JI (Fall 2024)

Non-programming exercises:

- Write in a neat and legible handwriting
- Clearly explain the reasoning process
- Write in a complete style (subject, verb and object)

Progamming exercises:

- Write a single README file per homework
- Push to git and create a release with tag h7

ECE4821: submit together with ECE4820

ECE4820 Exercises

Ex. 1 — *Page replacement algorithm*

In this exercise we consider the WSClock page replacement algorithm with a τ value of two ticks. The system state is given as follows.

Page	Time stamp	Present	Referenced	Modified
0	6	1	0	1
1	9	1	1	0
2	9	1	1	1
3	7	1	0	0
4	4	0	0	0

1. Explain the content of the new table entries if a clock interrupt occurs at tick 10.
2. Due to a read request to page 4 a page fault occurs at tick 10. Describe the new table entry.

Ex. 2 — *Tracking subprocesses using eBPF*

As a prerequisite, make sure to install BPF Compiler Collection (BCC) toolkit.

In this exercise, we want to write an eBPF program to track the number of sub-processes forked from it.

1. Which system call should be traced?
2. Write an eBPF program based on kprobes and/or kretprobes to implement the task.

   Hint. Quickly search what Kprobes is and how to use it with eBPF.
3. What is `fentry` and how does it differ from Kprobes? How could it be used to complete the task?

Ex. 3 — *Research*

Write about a page on the topic of the `ext2` filesystem. Do not forget to reference your sources.

Ex. 4 — *Simple questions*

1. If a page is shared between two processes, is it possible that the page is read-only for one process and read-write for the other? Why or why not?
2. When both paging and segmentation are being used, first the segment descriptor is found and then the page descriptor. Does the TLB also need a two-levels lookup?

ECE4821 Exercises

Ex. 5 — *Simple question*

A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the address space? If the page size were 512 bytes, would it fit?

Ex. 6 — *Research*

Write about half a page on the topic of the `minix` filesystem. Do not forget to reference your sources.

Ex. 7 — *Tracking subprocesses using eBPF, again!*

The Linux kernel provides trace-points which allow to attach at points other than functions.

1. Looking at `/sys/kernel/tracing/available_events`, which event may be useful to track the number of sub-processes?

   Hint. Restrict your attention to lines with the `sched` keyword.

2. What is should the data structure for the trace-point look like?

   Hint. Search `/sys/kernel/debug/tracing/events/sched/yourselectedtracepoint/format`.

3. Rewrite your program from exercise 2 to use a trace-point instead of a system call.